IS 6733
Deep Learning on Cloud Platforms

Lecture 4a

Machine Learning Fundamentals – Part1

Dr. Yuanxiong Guo

Department of Information Systems and Cyber Security

Acknowledgement

- Some slides are adapted from Professor Widom's Instructional Odyssey
    - www.professorwidom.org

- Thanks to Prof. Jennifer Widom for sharing the slides

Machine Learning Definition

- A machine Learning algorithm is an algorithm that is able to learn from data
  - Build models from data
  - The models have *tunable parameters* that can be adapted to observed data
  - Once these models have been fit to previously seen data, they can be used to predict and understand aspects of newly observed data

3 Types of Learning

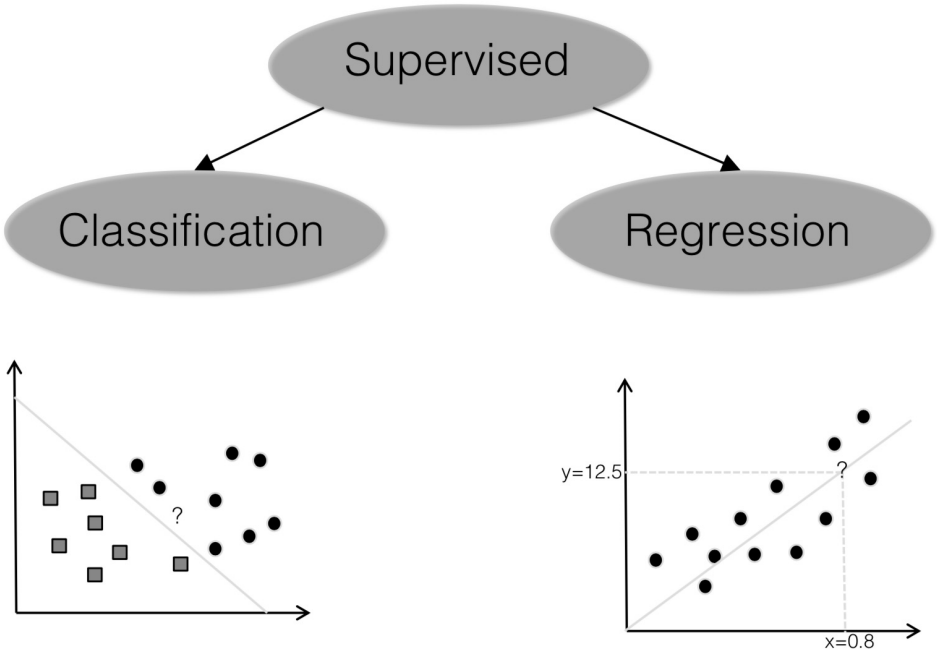
- Learning from labeled data
- E.g., Spam classification

- Discover structure in unlabeled data
- E.g., Document clustering

- Learning by "doing" with delayed reward
- E.g., Chess computer

Supervised Learning

- Given examples of a function (X, F(X))
- Predict function F(x) for new examples X
  - Discrete F(X): Classification
  - Continuous F(X): Regression



Regression and Classification Examples

- Stock prediction
  - Predict the price of a stock (y)
  - Depends on x =
    - Recent history of stock price
    - News event
    - Related commodities

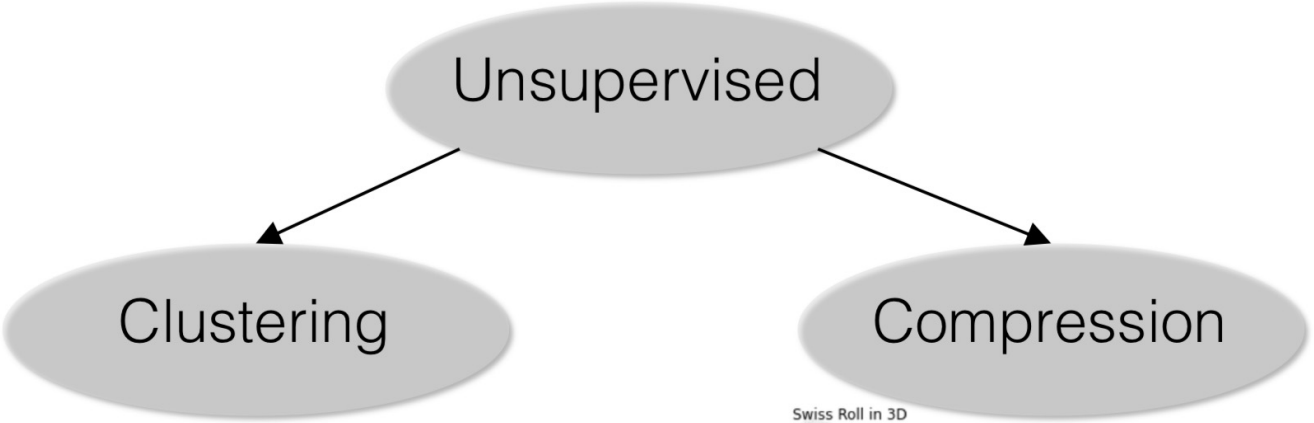
- Spam or Not Spam emails

- Music or Tweeter Sentiment Analysis

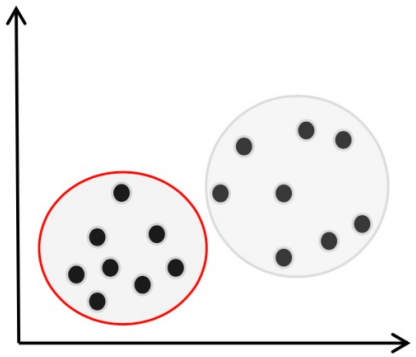
More Supervised Learning Examples

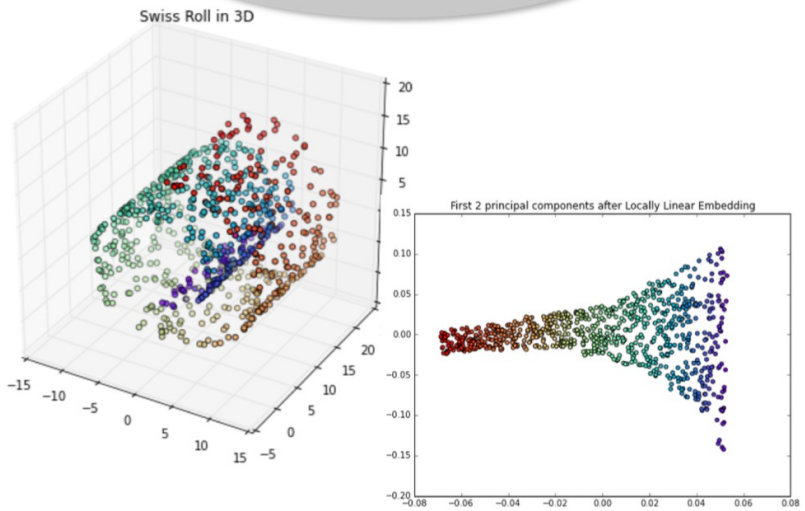
- Sequence generation
  - Given a picture, predict a caption describing it
- Syntax tree prediction
  - Given a sentence, predict its decomposition into a syntax tree
- Object detection
  - Given a picture, draw a bounding box around certain objects inside the picture
- Image segmentation
  - Given a picture, draw a pixel-level mask on a specific object

Unsupervised Learning



Dimensionality Reduction





Reinforcement Learning (RL)

- In RL, an agent receives information about its environment and learns to choose actions that will maximize some reward
  - E.g., a neural network that looks at a video-game screen and outputs game actions to maximize its score
- Currently, RL is mostly a research area and hasn't yet had significant practical successes beyond games
- In the future, expect to see more RL applications
  - Self-driving cars, robotics, resource management, education, and so on.

Regression

Regression

Using data to build models and make predictions

- Supervised
- Training data, each example:
  - Set of predictor values – “independent variables”
  - Numeric output value – “dependent variable”
- Model is function from predictors to output
  - Use model to predict output value for new predictor values
- Example
  - Predictions: mother height, father height, current age
  - Output: height

Regression

- Set of predictor values – "independent variables"
- Numeric output value – "dependent variable"
- Model is function from predictors to output

Training data

$w_1, x_1, y_1, z_1 \rightarrow o_1$
$w_2, x_2, y_2, z_2 \rightarrow o_2$
$w_3, x_3, y_3, z_3 \rightarrow o_3$
……

Model

$f(w, x, y, z) = o$

Regression

- Goal: Function *f* applied to training data should produce values as close as possible in aggregate to actual outputs

Training data

$w_1, x_1, y_1, z_1 \rightarrow o_1$
$w_2, x_2, y_2, z_2 \rightarrow o_2$
$w_3, x_3, y_3, z_3 \rightarrow o_3$
……

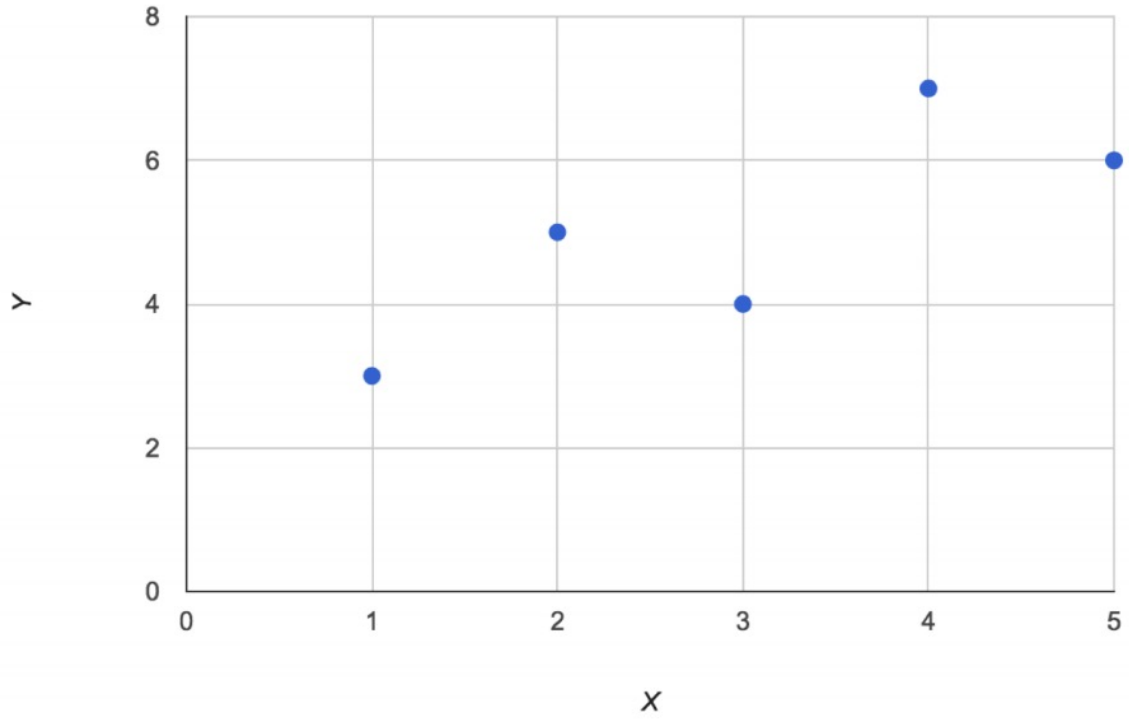
Model

$f(w, x, y, z) = o$

$f(w_1, x_1, y_1, z_1) = o_1'$
$f(w_2, x_2, y_2, z_2) = o_2'$
$f(w_3, x_3, y_3, z_3) = o_3'$

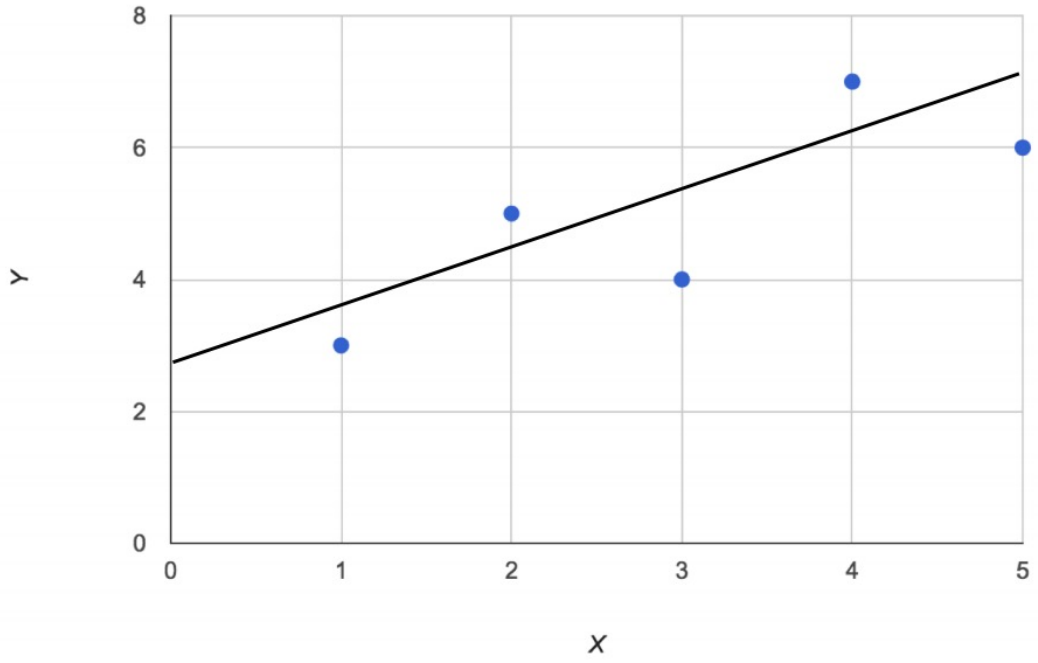
Simple Linear Regression

- We will focus on
  - One numerical predictor value, call it *x*
  - One numerical output value, call it *y*
- Data items are points in two-dimensional space



Simple Linear Regression

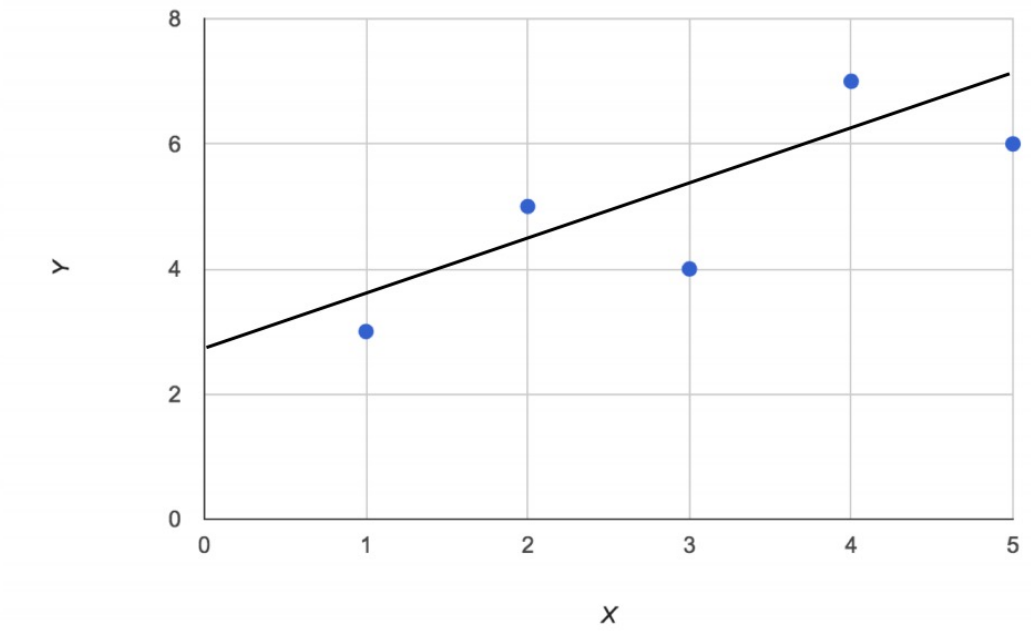
- We will focus on
  - One numerical predictor value, call it *x*
  - One numerical output value, call it *y*
  - Functions *f(x) = y* that are lines (for now)



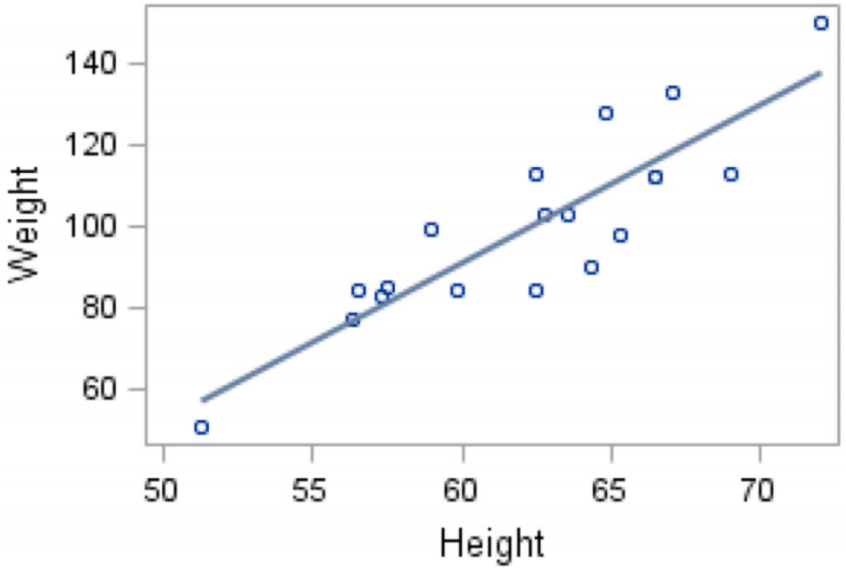
Simple Linear Regression

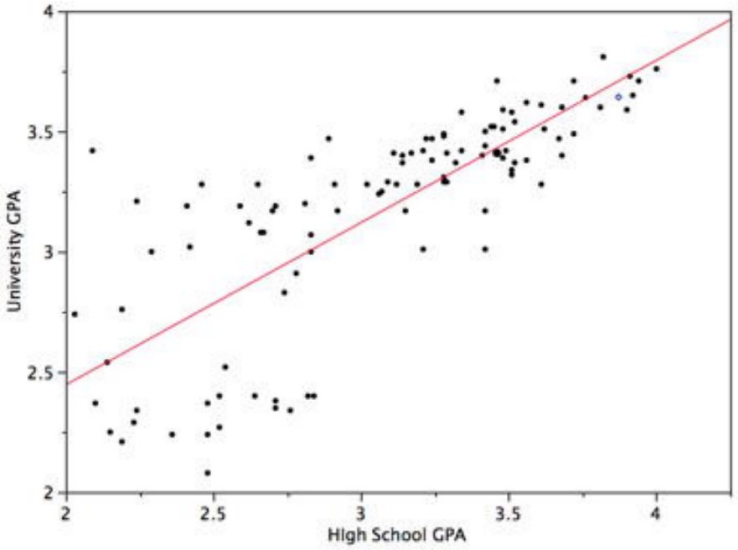
Functions $f(x) = y$ that are lines: $y = a\,x + b$

$$y = 0.8\,x + 2.6$$



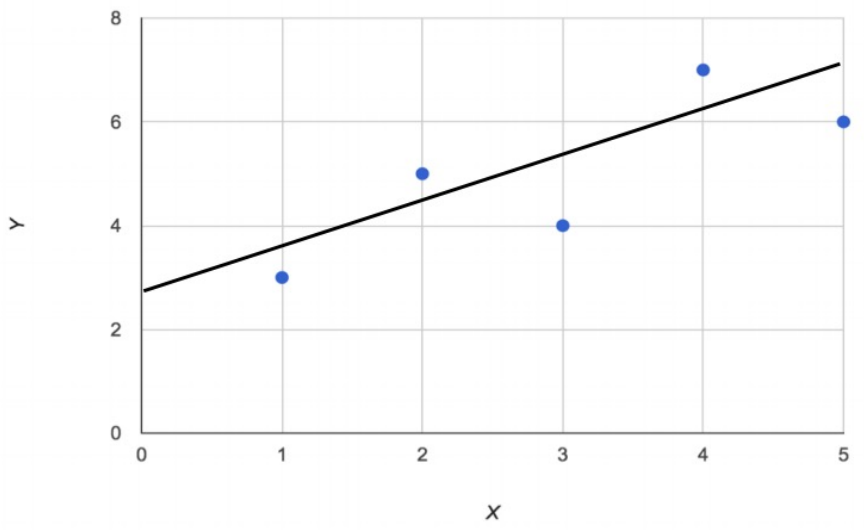
Real Examples





Summary So Far

- Given: Set of known (x, y) points
- Find: function $f(x) = a\,x + b$ that "best fits" the known points, i.e., $f(x)$ is close to y
- Use function to predict y values for new x's
  - Also can be used to test correlation



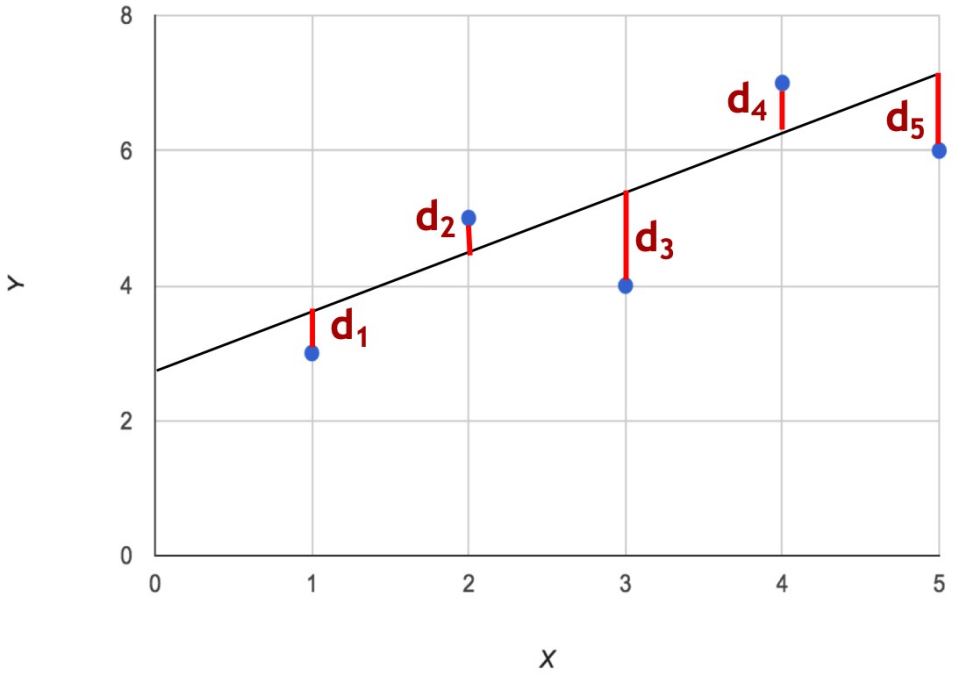
Calculating Simple Linear Regression

Method of least squares

- Given a point and a line, the error for the point is its vertical distance d from the line, and the squared error is d^2
- Given a set of points and a line, the sum of squared error (SSE) is the sum of the squared errors for all the points
- Goal: Given a set of points, find the line that minimizes the SSE

Calculating Simple Linear Regression

Method of least squares

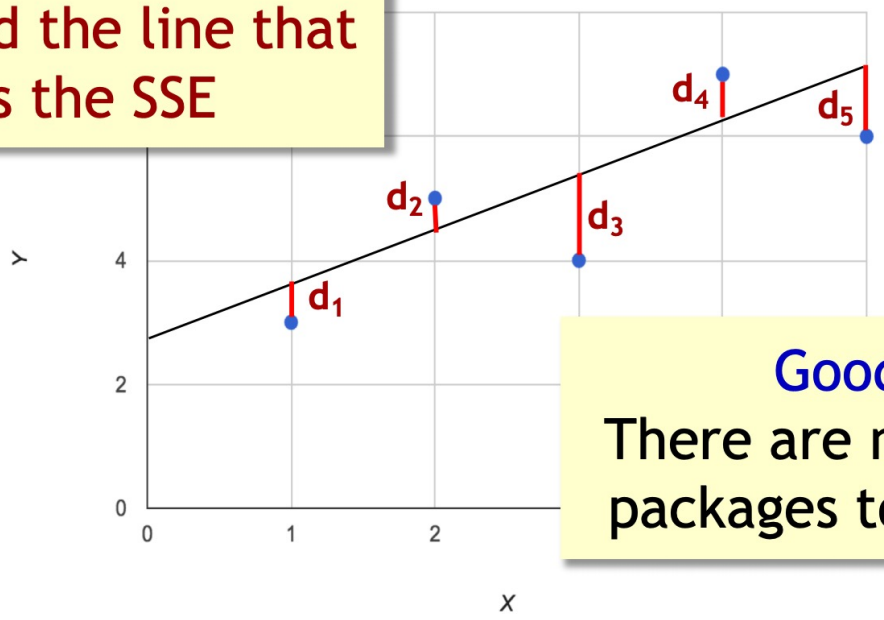


$$SSE = d_1^2 + d_2^2 + d_3^2 + d_4^2 + d_5^2$$

Calculating Simple Linear Regression

Method of least squares

Goal: Find the line that minimizes the SSE



Good News!
There are many software packages to do it for you

$$SSE = d_1^2 + d_2^2 + d_3^2 + d_4^2 + d_5^2$$

Calculating Simple Linear Regression

- For regression algorithms, three evaluation metrics are commonly used:
  - Mean Absolute Error (MAE): the mean of the absolute value of the errors

$$MAE = \frac{1}{n}\sum_{j=1}^{n}|y_j - \hat{y}_j|$$

  - Mean Squared Error (MSE): the mean of the squared errors

$$MSE = \frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2$$

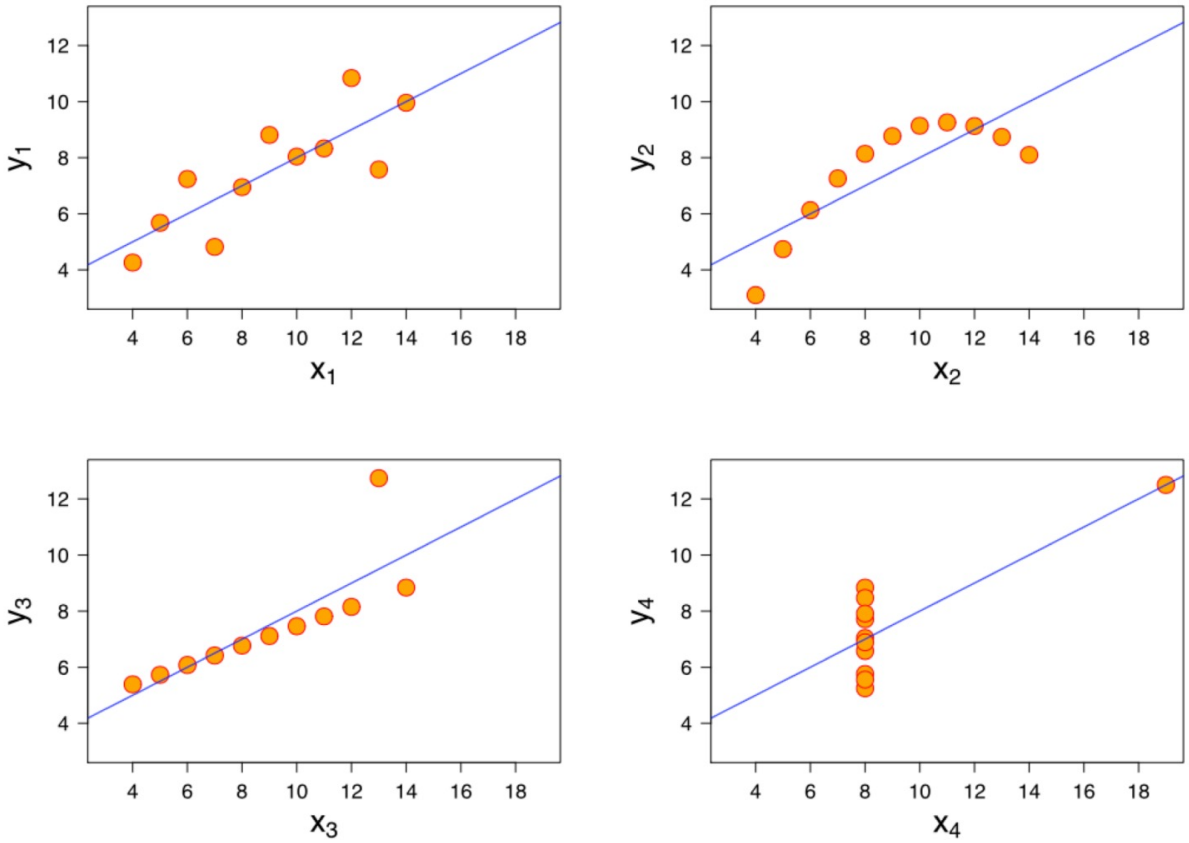
  - Root Mean Squared Error (RMSE): the square root of the mean of the squared errors

$$RMSE = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2}$$

Shortcomings of Simple Linear Regression

Anscombe's Quartet (From Overview)

Also identical R^2 values!



Remainder

Goal: Function f applied to training data should produce values as close as possible in aggregate to actual outputs

Training data

$w_1, x_1, y_1, z_1 \rightarrow o_1$
$w_2, x_2, y_2, z_2 \rightarrow o_2$
$w_3, x_3, y_3, z_3 \rightarrow o_3$
……

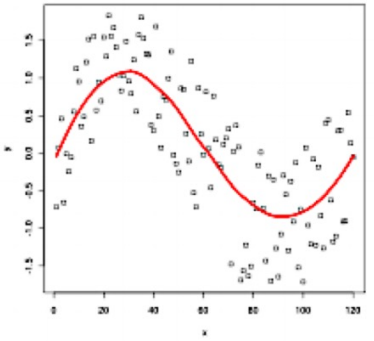
Model

$f(w, x, y, z) = o$

$f(w_1, x_1, y_1, z_1) = o_1'$
$f(w_2, x_2, y_2, z_2) = o_2'$
$f(w_3, x_3, y_3, z_3) = o_3'$

Polynomial Regression

Given: Set of known (x,y) points
Find: function f that "best fits" the known points, i.e., $f(x)$ is close to y



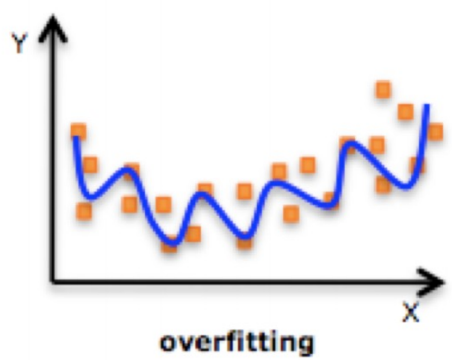
$$f(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n$$

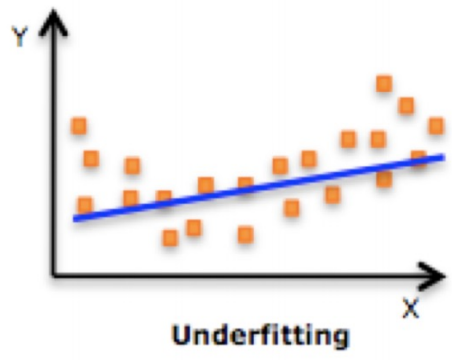
- "Best fit" is still method of least squares
- Still have coefficient of determination R^2 (no r)
- Pick smallest degree ***n*** that fits the points reasonably well

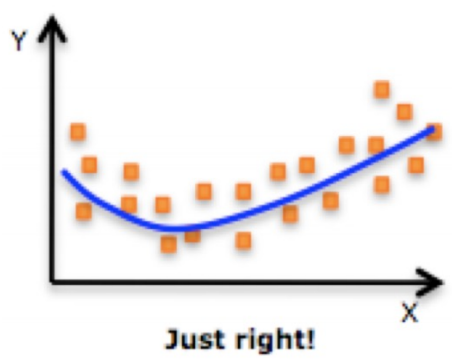
Also exponential regression: $f(x) = a\, b^x$

Dangers of (Polynomial) Regression

Overfitting and Underfitting (From Overview)







Regression Summary

- Supervised machine learning
- Training data
  - Set of input values with numeric output value
- Model is function from input to output
  - Use function to predict output value for inputs
- Balance complexity of function against “best fit”

Classification

Classification

Using data to build models and make predictions

- Supervised
- Training data, each example:
  - Set of feature values – numeric or categorical
  - Categorical output value – "label"
- Model is method from feature values to label
  - Use model to predict label for new feature values
- Example
  - Feature values: age, gender, income, profession
  - Label: buyer, non-buyer

Other Examples

- Medical diagnosis
  - Feature values: age, gender, history, symptom1-severity, symptom2-severity, test-result1, test-result2
  - Label: disease
- Email spam detection
  - Feature values: sender-domain, length, #images, keyword_1, keyword_2, …, keyword_n
  - Label: spam or not-spam
- Credit card fraud detection
  - Feature values: user, location, item, price
  - Label: fraud or okay

Algorithms for Classification

- Despite similarity of problem statement to regression, non-numerical nature of classification leads to completely different approaches
  - K-nearest neighbors (k-NN)
  - Decision trees
  - Naïve Byes
  - … and others

K-Nearest Neighbors (KNN)

- For any pair of data item i_1 and i_2, from their feature values compute *distance*(i_1, i_2)
- Example:
  - Features: gender, profession, age, income, postal-code
    - $person_1$ = (male, teacher, 47, $25K, 94305)
    - $person_2$ = (female, teacher, 43, $28K, 94309)
    - distance($person_1$, $person_2$)
  - distance() can be defined as inverse of similarity()

K-Nearest Neighbors (KNN)

- Features: gender, profession, age, income, postal-code
  - $person_1$ = (male, teacher, 47, $25K, 94305) buyer
  - $person_2$ = (female, teacher, 43, $28K, 94309) non-buyer

- Remember training data has labels

To classify a new item *i*: In the labeled data, find the K closest items to *i*, assign most frequent label

$person_3$ = (female, doctor, 40, $40K, 95123)

KNN Example

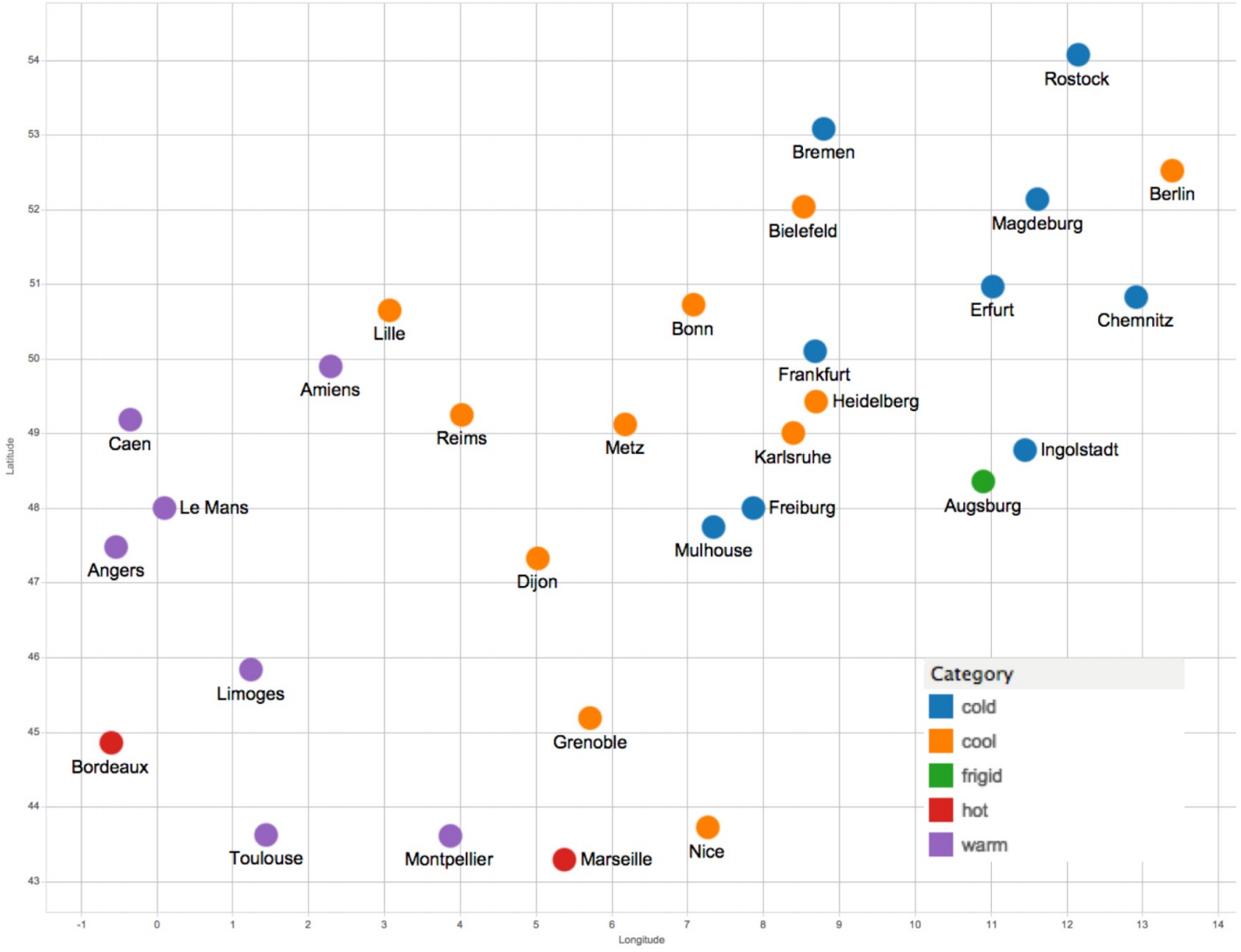
- City temperatures – France and Germany
- Features: longitude, latitude
- Distance is Euclidean distance
  - distance($[o_1, a_1]$, $[o_2, a_2]$) = sqrt($(o_1 - o_2)^2 + (a_1 - a_2)^2$)

    = actual distance in x-y plane
- Labels: frigid, cold, cool, warm, hot

Nice (7.27, 43.72) cool
Toulouse (1.45, 43.62) warm
Frankfurt (8.68, 50.1) cold
……

Predict temperature category from longitude and latitude

KNN Example


Rostock
Bremen
Berlin
Magdeburg
Bielefeld
Erfurt
Chemnitz
Lille
Bonn
Frankfurt
Amiens
Heidelberg
Caen
Reims
Metz
Karlsruhe
Ingolstadt
Le Mans
Augsburg
Freiburg
Mulhouse
Angers
Dijon
Limoges
Grenoble
Bordeaux
Toulouse
Montpellier
Marseille
Nice
Category
cold
cool
frigid
hot
warm
Latitude
Longitude
54
53
52
51
50
49
48
47
46
45
44
43
-1
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14

KNN Summary

To classify a new item *i*: find the K closest items to *i* in the labeled data, assign most frequent label

- No hidden complicated math!
- Once distance function is defined, rest is easy
- Though not necessarily efficient
  - Real examples often have thousands of features
    - Medical diagnosis: symptoms (yes/no), test results
    - Email spam detection: words (frequency)
  - Database of labeled items might be enormous

"Regression" Using KNN

- Features: gender, profession, age, income, postal-code
  - $person_1$ = (male, teacher, 47, $25K, 94305) buyer
  - $person_2$ = (female, teacher, 43, $28K, 94309) non-buyer

- Remember training data has labels

To classify a new item *i*: find the K closest items to *i* in the labeled data, assign most frequent label

$person_3$ = (female, doctor, 40, $40K, 95123)

"Regression" Using KNN

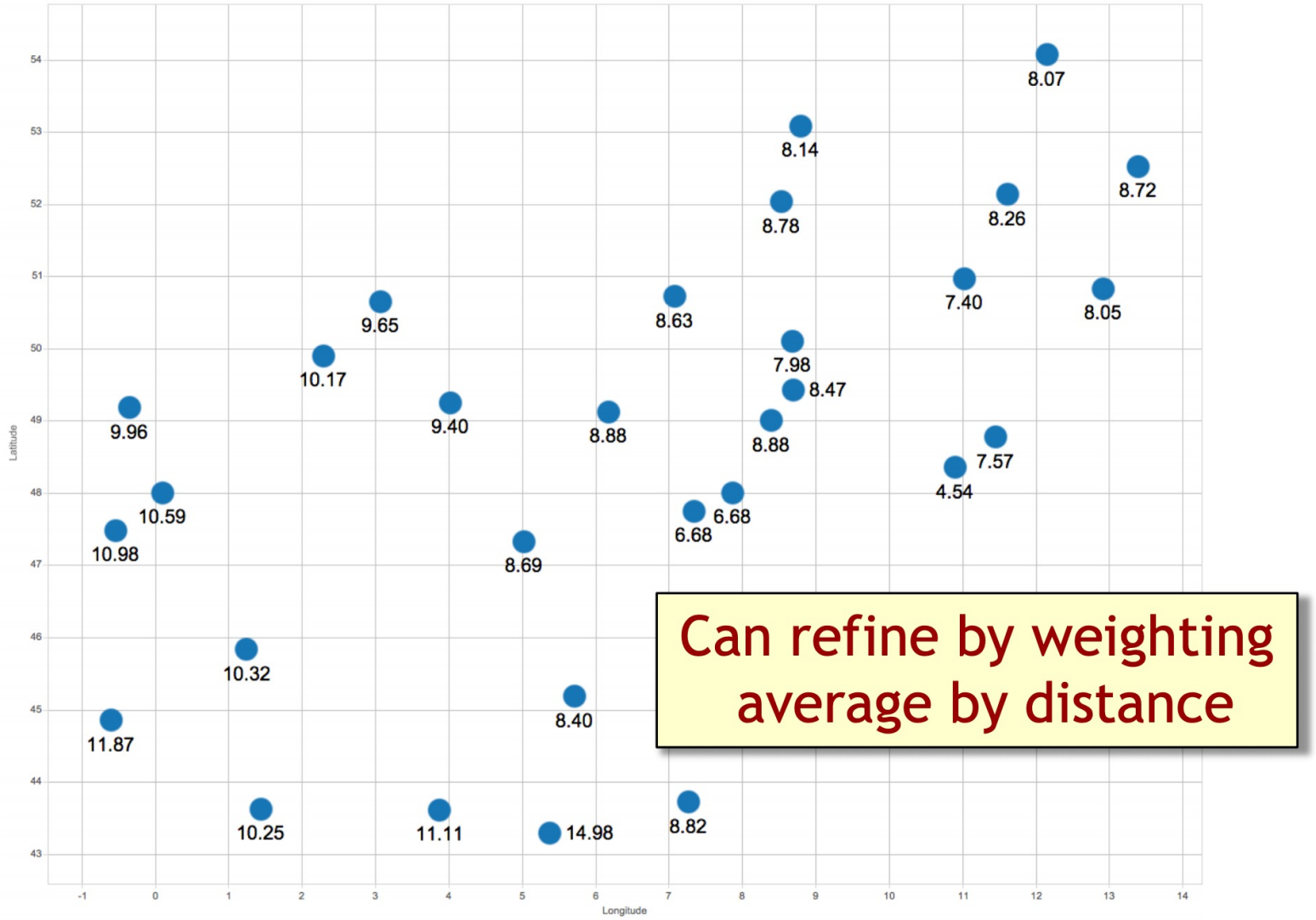
- Features: gender, profession, age, income, postal-code
  - $person_1$ = (male, teacher, 47, $25K, 94305) $250
  - $person_2$ = (female, teacher, 43, $28K, 94309) $100

- Remember training data has labels

To classify a new item *i*: find the K closest items to *i* in the labeled data, assign average value of labels

$person_3$ = (female, doctor, 40, $40K, 95123)

Regression Using KNN - Example

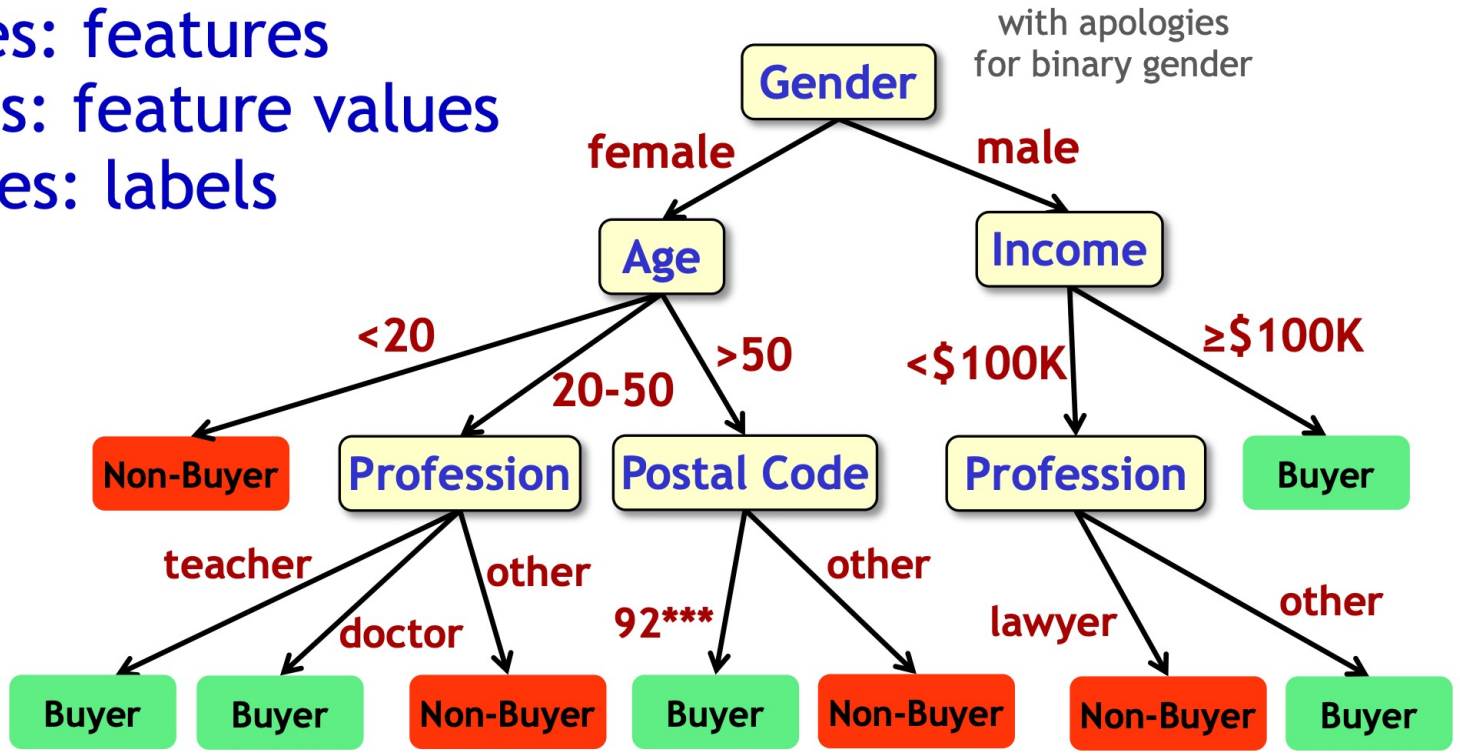


Decision Trees

- Use the training data to construct a decision tree
- Use the decision tree to classify new data

Decision Trees

Nodes: features
Edges: feature values
Leaves: labels



New data item to classify:
Navigate tree based on feature values

Decision Trees

- Primary challenges is building good decision trees from training data
  - Which features and feature values to use at each choice point
  - HUGE number of possible trees even with small number of features and values
- Common approach: “forest” of many trees, combine the results
  - Still impossible to consider all trees

Naïve Bayes

- Given new data item *i*, based on *i's* feature values and the training data, compute the probability of each possible label. Pick highest one.
- Efficiency relies on conditional independence assumption:

> Given any two features F_1, F_2 and a label L, the probability that $F_1 = v_1$ for an item with label L is independent of the probability that $F_2 = v_2$ for that item

- Examples:
  - gender and age? income and postal code?

Naïve Bayes

- Given new data item *i*, based on *i's* feature values and the training data, compute the probability of each possible label. Pick highest one.
- Efficiency relies on conditional independence assumption:

Conditional independence assumption often doesn't hold, which is why the approach is "naive"

abel L, the probability independent of the

Nevertheless the approach works very well in practice

- Examples:
  - gender and age? income and postal code?

Naïve Bayes Example

- Predict temperature category for a country based on whether the country has coastline and whether it is in the EU

country	coastline	EU	tempAvg	category
Albania	yes	no	15.18	hot
Andorra	no	no	9.60	warm
Belarus	no	no	5.95	cool
Belgium	yes	yes	9.65	warm
Bosnia and Herzegov	no	no	9.60	warm
Bulgaria	yes	yes	10.44	warm
Croatia	yes	yes	10.87	warm
Czech Republic	no	yes	7.86	cool
Denmark	yes	yes	7.63	cool
Estonia	yes	yes	4.59	cold
Finland	yes	yes	3.49	cold
Germany	yes	yes	7.87	cool
Greece	yes	yes	16.90	hot
Hungary	no	yes	9.60	warm
Ireland	yes	yes	9.30	warm

Naïve Bayes Preparation

- Step 1: Compute fraction (probability) of items in each category

cold	.18
cool	.38
warm	.24
hot	.20

Naïve Bayes Preparation

- Step 2: For each category, compute fraction of items in that category for each feature and value

cold (.18)	coastline=yes	.83
	coastline=no	.17
	EU=yes	.67
	EU=no	.33
cool (.38)	coastline=yes	.69
	coastline=no	.31
	EU=yes	.77
	EU=no	.23

warm (.24)	coastline=yes	.5
	coastline=no	.5
	EU=yes	.5
	EU=no	.5
hot (.20)	coastline=yes	1.0
	coastline=no	.0
	EU=yes	.71
	EU=no	.29

Naïve Bayes Prediction

- New item: France, coastline = yes, EU = yes
- For each category: probability of category times product of probabilities of new item's features in that category. Pick highest!

category	prob.	coastline=yes	EU=yes	product
cold	.18	.83	.67	.10
cool	.38	.69	.77	.20
warm	.24	.5	.5	.06
hot	.20	1.0	.71	.14

Naïve Bayes Prediction

- New item: Serbia, coastline = no, EU = no
- For each category: probability of category times product of probabilities of new item's features in that category. Pick highest!

category	prob.	coastline=no	EU=no	product
cold	.18	.17	.33	.01
cool	.38	.31	.23	.03
warm	.24	.5	.5	.06
hot	.20	.0	.29	.00

Naïve Bayes Prediction

- New item: Austria, coastline = no, EU = yes
- For each category: probability of category times product of probabilities of new item's features in that category. Pick highest!

category	prob.	coastline=no	EU=yes	product
cold	.18	.17	.67	.02
cool	.38	.31	.77	.09
warm	.24	.5	.5	.06
hot	.20	.0	.71	.0

Naïve Bayes Prediction

- New item: Austria, coastline = no, EU = yes
- For
pro
cat

Many presentations of Naïve Bayes include an additional normalization step so the final products are probabilities that sum to 1.0. The choice of label is unchanged, so we've omitted that step for simplicity.

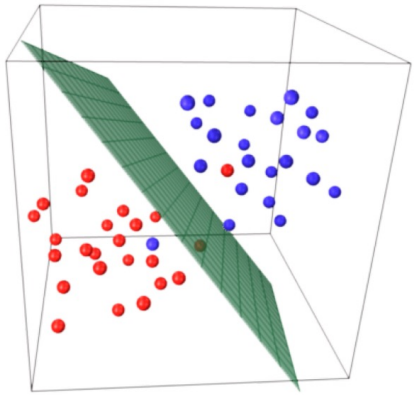
that

ca				
cold	.18	.17	.67	.02
cool	.38	.31	.77	.09
warm	.24	.5	.5	.06
hot	.20	.0	.71	.0

Other Terms You Might Hear

- Logistic regression
  - Recall regression model is function *f* from predictor values to numeric output value
  - For classification: from training data obtain one regression function f_L for each label *L*

    f_L(feature-values) = probability for item having label *L*
- Support Vector Machine
  - Two labels only ("binary classifier")
  - Features = multidimensional space
  - From training data SVM finds hyper-plane that best divides space according to labels



Classification Summary

- Supervised machine learning
- Training data, each example:
  - Set of feature values – numeric or categorical
  - Categorical output value – label
- Model is ”function” from feature values to label
  - Use model to predict label for new feature values
- Approaches we covered
  - K-nearest neighbors: relies on distance (or similarity) function
  - Decision trees: relies on finding good trees/forests
  - Naïve Bayes: relies on conditional independence assumption

Clustering

Clustering

Like classification, data items consist of values for a set of features (numeric or categorical)

- Medical patients
  - Feature values: age, gender, symptom1-serverity, symptom2-serverity, test-result1, test-result2
- Web pages
  - Feature values: URL domain, length, #images, $heading_1$, $heading_2$, …., $heading_n$
- Products
  - Feature values: category, name, size, weight, price

Unlike classification, there is no label

Clustering

Like K-NN, for any pair of data items i_1 and i_2, from their feature values can compute distance function: *distance*(i_1, i_2)

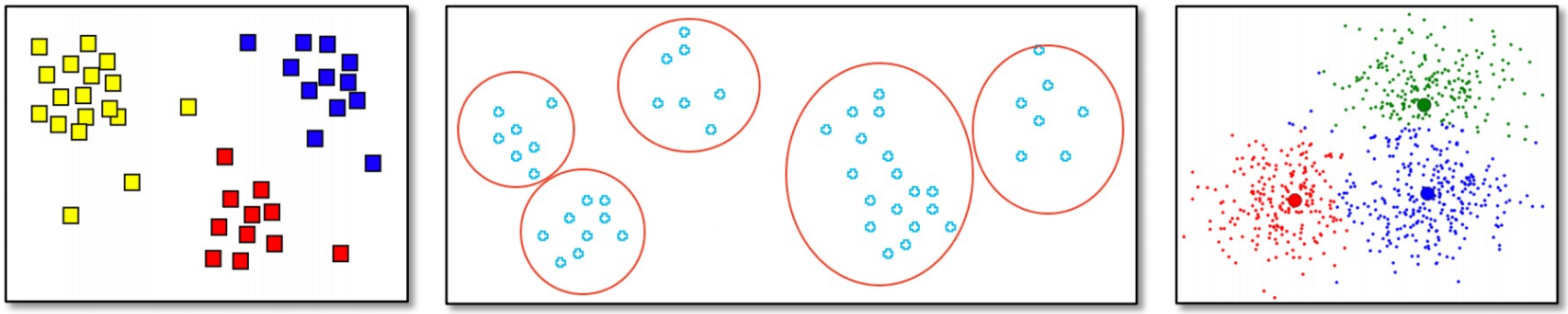
- Example
  - Features: gender, profession, age, income, and postal-code
    - $person_1$ = (male, teacher, 47, $25K, 94305)
    - $person_2$ = (female, teacher, 43, $28K, 94309)
    - distance($person_1$, $person_2$)

distance() can be defined as inverse of *similarity()*

Clustering

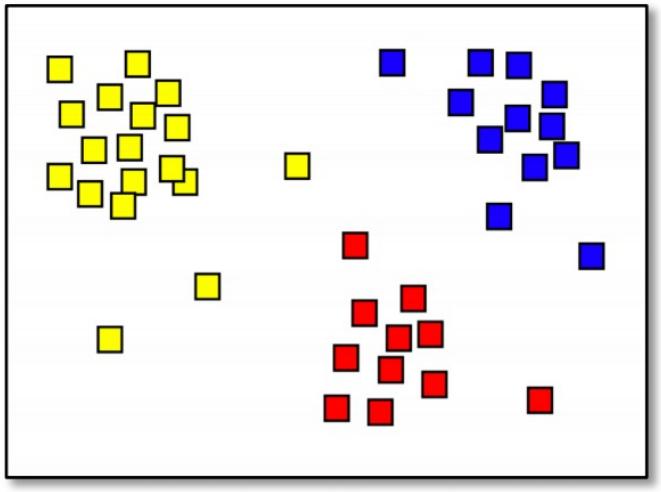
GOAL: Given a set of data items, partition them into groups (= clusters) so that items within groups are close to each other based on distance function

- Sometimes number of clusters is pre-specified
- Typically clusters need not be the same size



Some Uses for Clustering

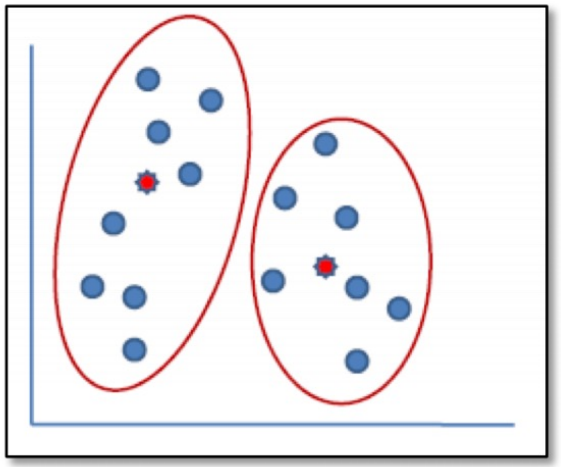
- Classification!
  - Assign labels to clusters
  - New data items get the label of their cluster
- Identify similar items
  - For substitutes or recommendation
  - For de-duplication
- Anomaly (outlier) detection
  - Items that are far from any cluster



K-Means Clustering

Reminder: for any pair of data items i_1 and i_2 have *distance*(i_1, i_2)

- For a group of items, the mean value (centroid) of the group is the item *i* (in the group or not) that minimizes the sum of *distance(i, i')* for all *i'* in the group

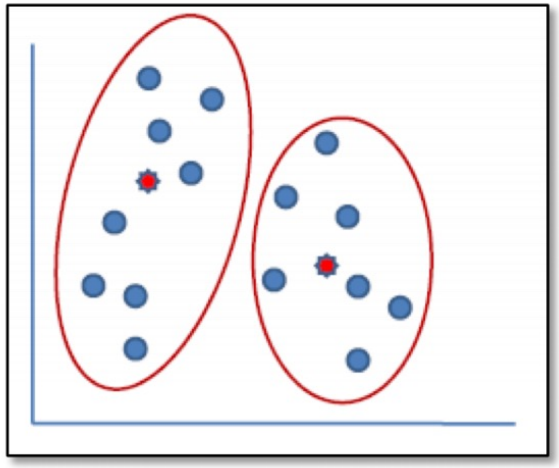


K-Means Clustering

For a group of items, the mean value (centroid) of the group is the item *i* (in the group or not) that minimizes the sum of *distance(i, i')* for all *i'* in the group

- Error for each item: distance *d* from the mean for its group; squared error is d^2
- Error for the entire clustering: sum of squared errors (SSE)

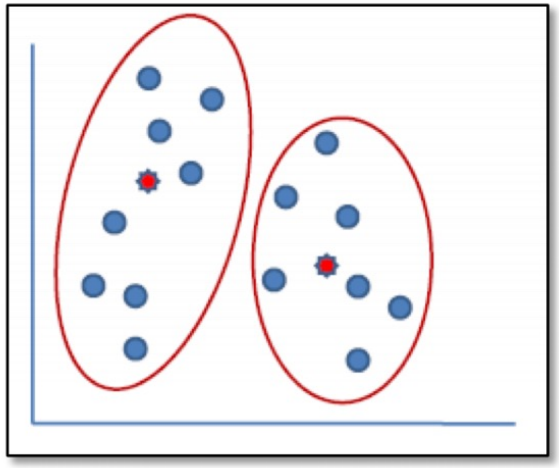
Remind you of anything?



K-Means Clustering

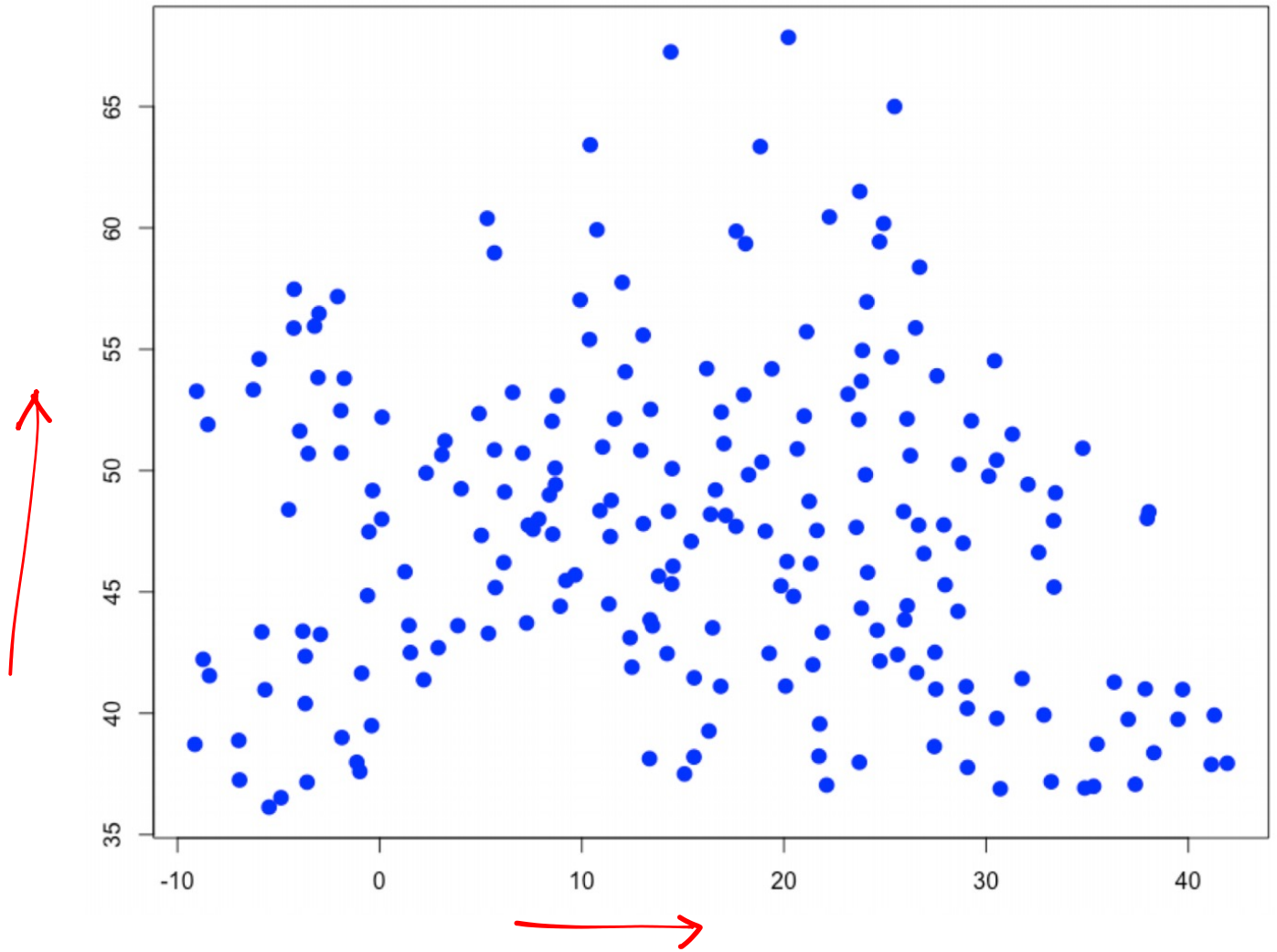
Given a set of data items and desired number of clusters *k*, K-means groups the items into *k* clusters minimizing the SSE

- Extremely difficult to compute efficiently
  - In fact, impossible
- Most algorithms compute an approximate solution (might not be absolute lowest SSE)



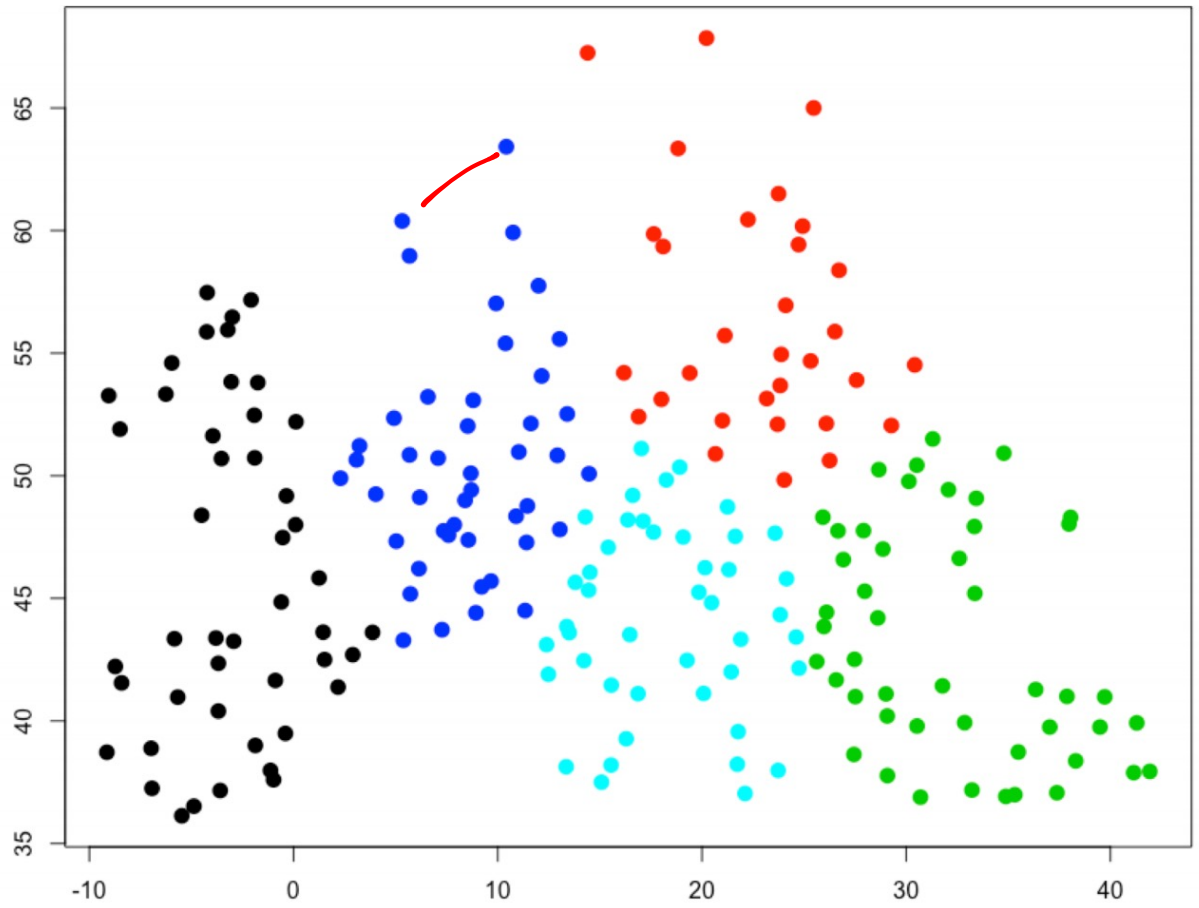
Clustering European Cities

By geographic distance, then by temperature



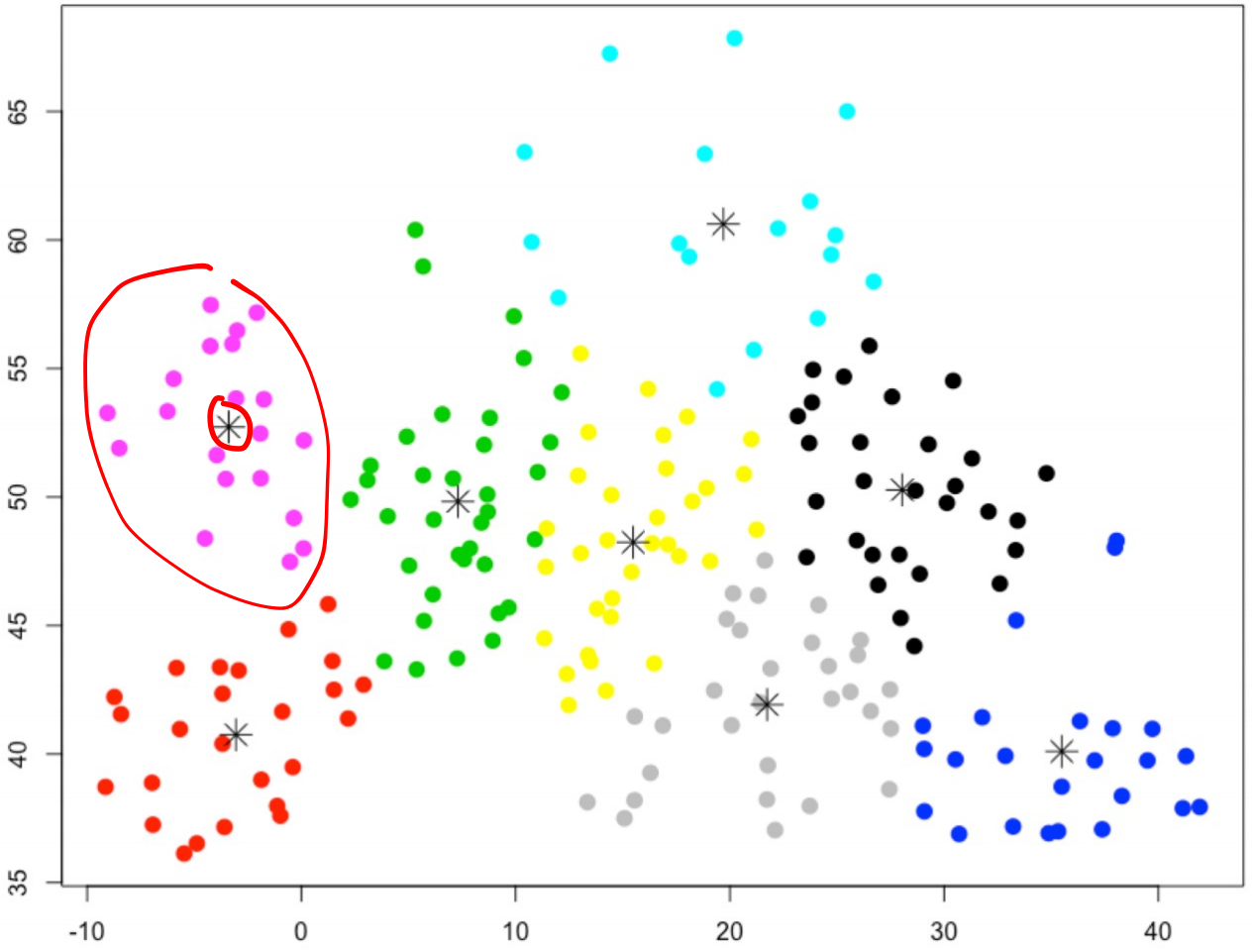
Clustering European Cities

Distance = actual distance, k = 5



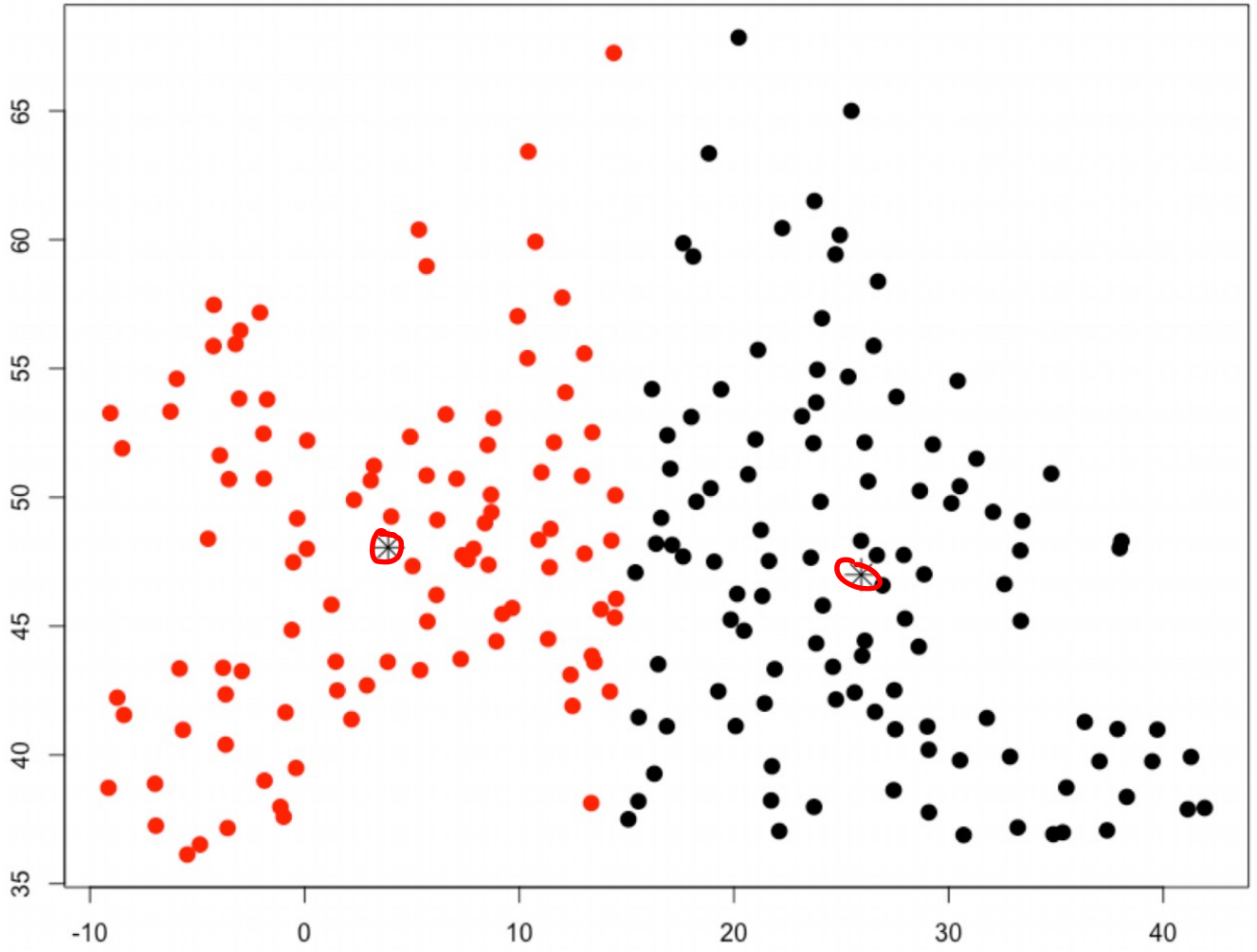
Clustering European Cities

Distance = actual distance, k = 8, with cluster means



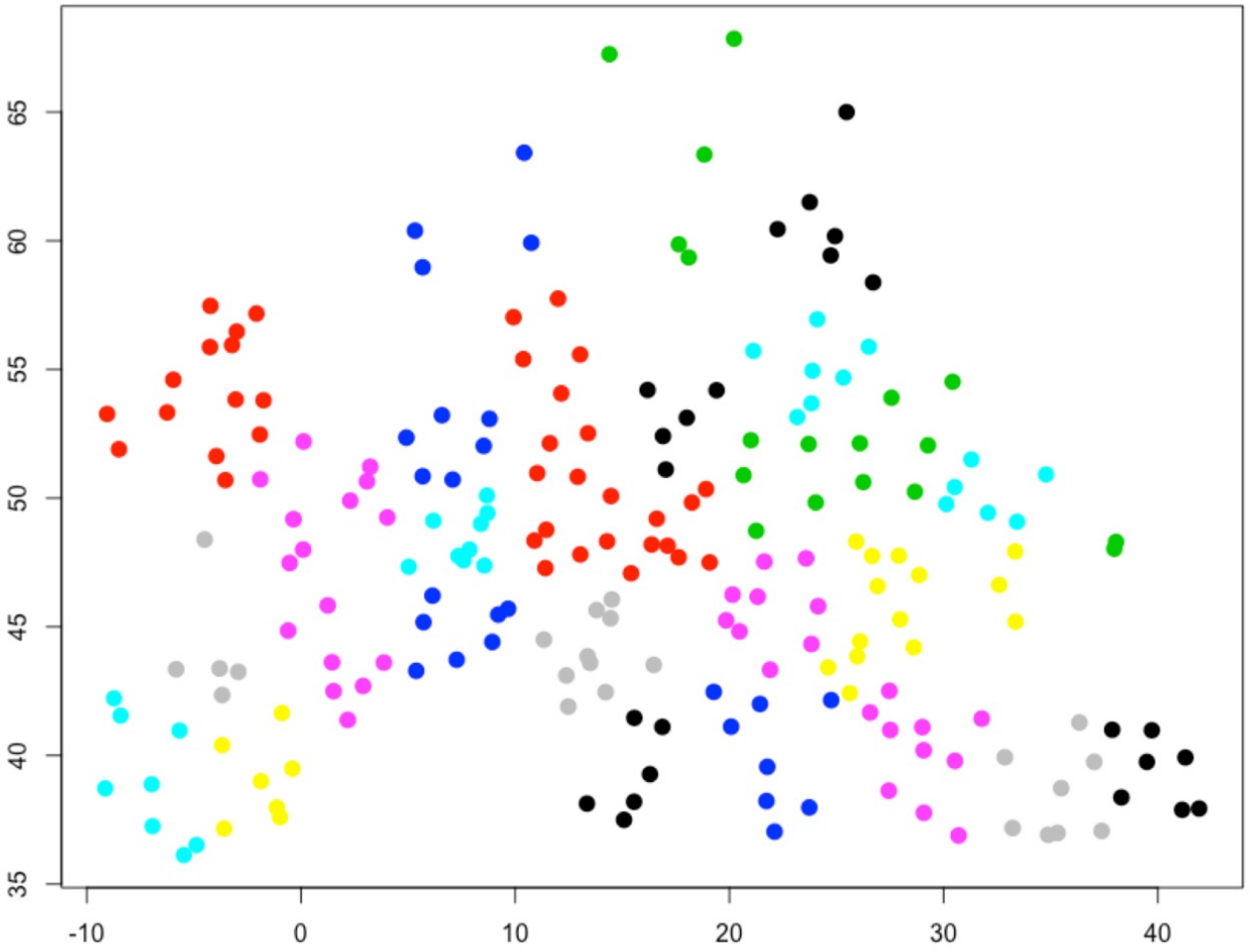
Clustering European Cities

Distance = actual distance, k = 2, with cluster means



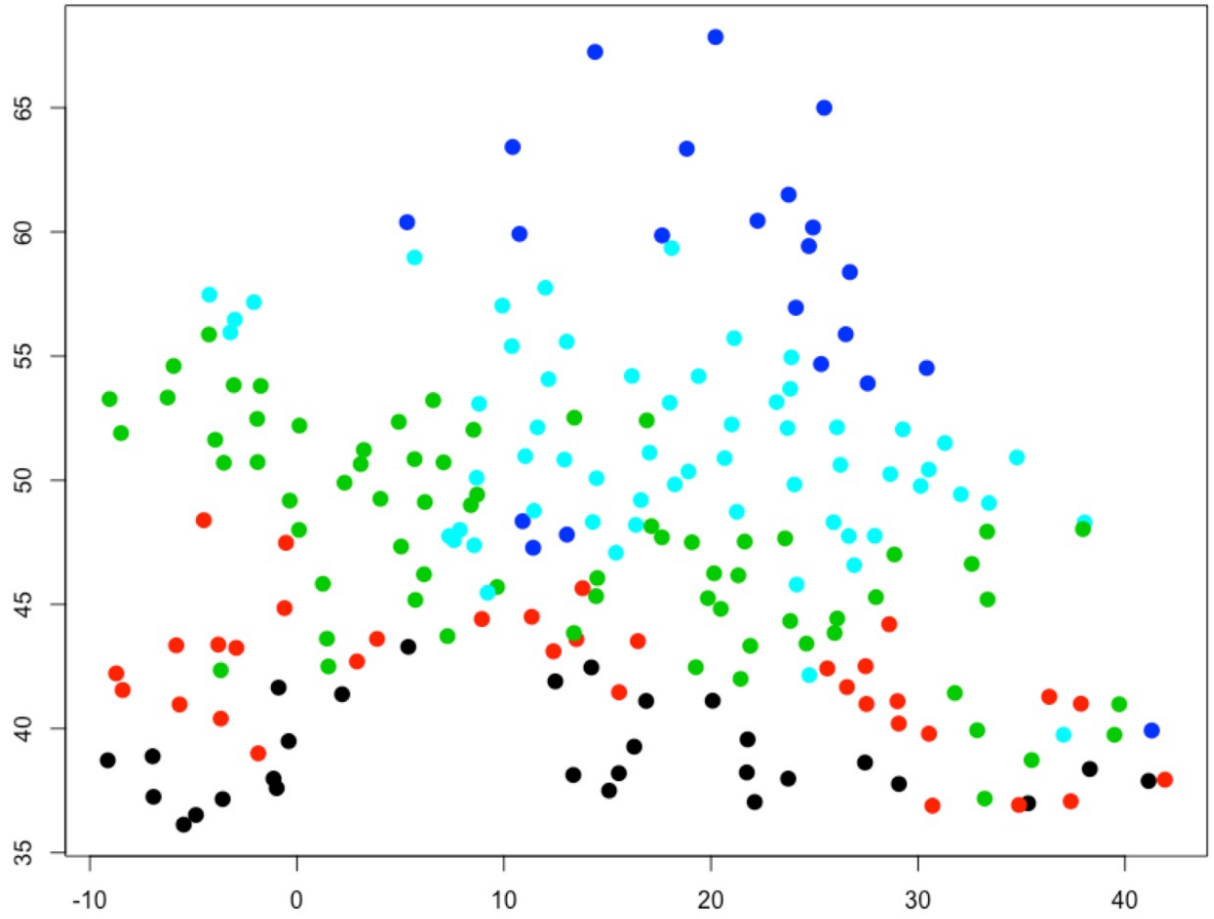
Clustering European Cities

Distance = actual distance , k = 30



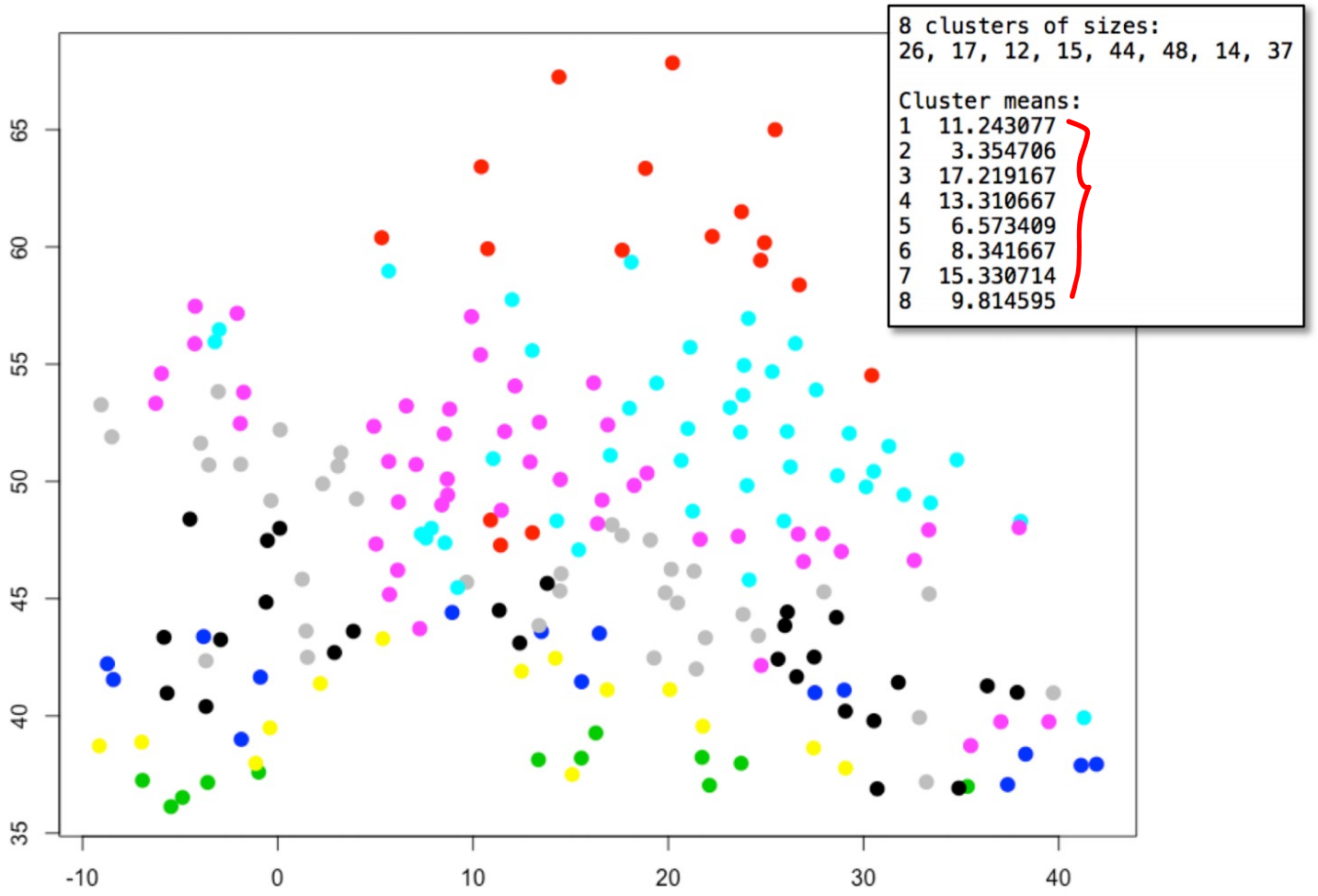
Clustering European Cities

Distance = temperature, k = 5



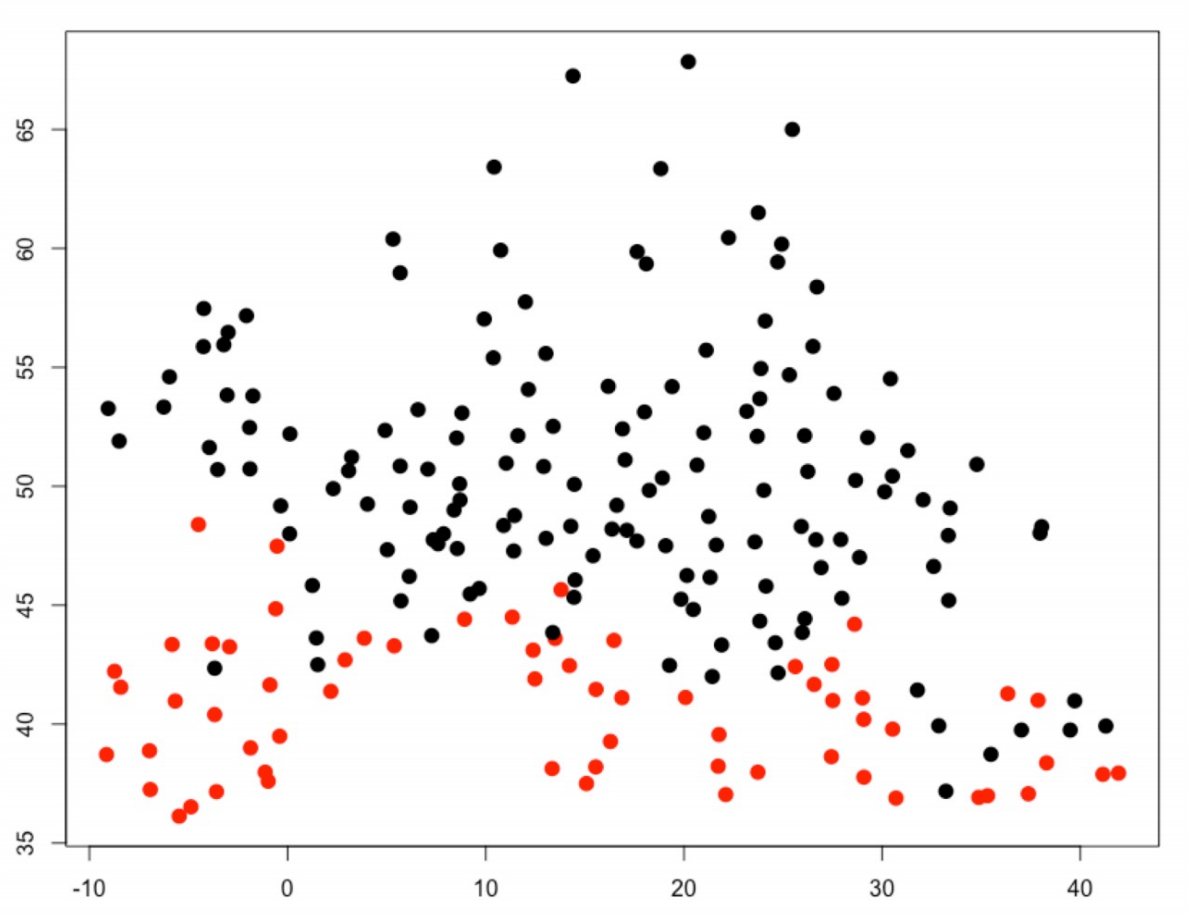
Clustering European Cities

Distance = temperature, k = 8, with means



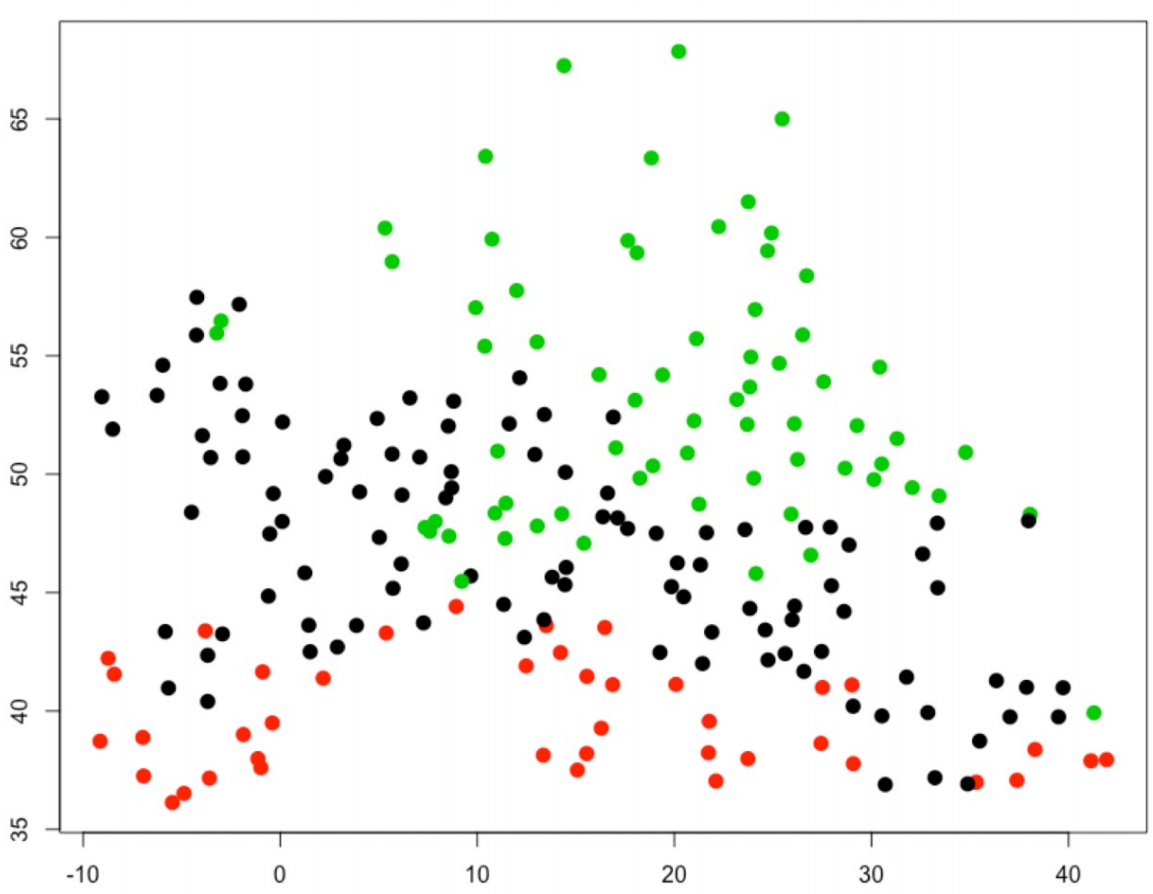
Clustering European Cities

Distance = temperature, k = 2



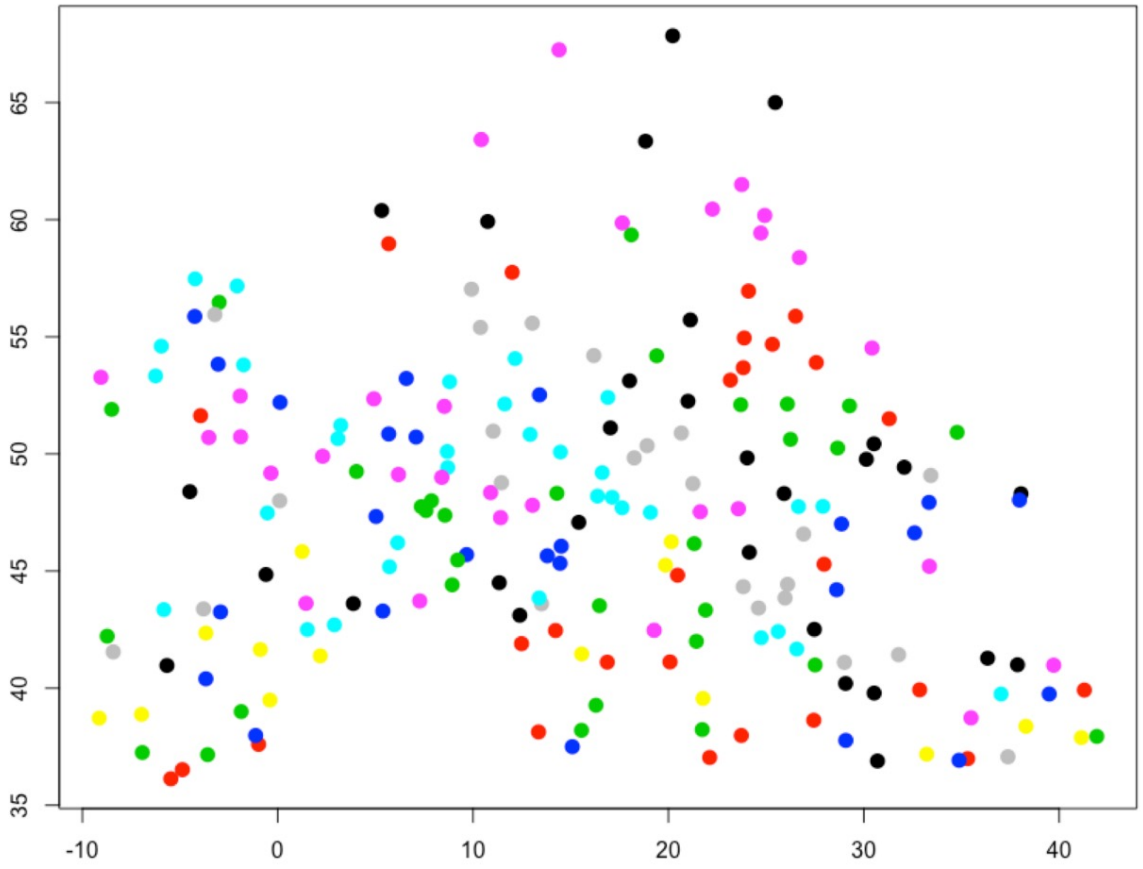
Clustering European Cities

Distance = temperature, k = 3



Clustering European Cities

Distance = temperature, k = 30



Scikit-Learn

- A Python package that provides efficient versions of a large number of common machine learning algorithms
- Data representation in Scikit-Learn
- Estimator API

Data Representation in Scikit-Learn

- Feature Matrix: X
  - Two-dimensional with shape [n_samples, n_features]
  - Samples (rows): individual objects described by the dataset
  - Features (columns): distinct observations that describe each sample in a quantitative manner
  - Contained in a NumPy array or Pandas DataFrame
- Label/Target Array: y
  - One-dimensional with shape [n_samples,]
  - Contained in a NumPy array or Pandas Series

Estimator API

Five steps in using Scikit-Learn estimator API

1. Choose a class of model by importing the appropriate estimator class from Scikit-Learn
2. Choose model hyperparameters by instantiating this class with desired values
3. Arrange data into a feature matrix and target vector following the discussion above
4. Fit the model to your data by calling the fit() method of the model instance
5. Apply the model to new data
   - For supervised learning, use the predict() method to predict labels for unknown data
   - For unsupervised learning, transform or infer properties of the data using the transform() or predict() method

Scikit-Learn Code Examples

- https://colab.research.google.com/drive/19EEF35gM51NqmBQ2nCp42JsJPTj4_MXh

Additional References

- Chapter 5 of [T1]